

LPORAN JOBSHEET 5

BRUTE FORCE DAN DIVIDE CONQUER

Mata Kuliah : Algoritma dan Struktur Data

Dosen : **Mungki Astiningrum, S.T., M.Kom.**



Ilham Dharma Atmaja

244107020220

Kelas : 1A

Absen : 14

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG TAHUN 2025

5.1 Tujuan Praktikum Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algoritma bruteforce dan divide-conquer

5.2 Kode Program

```
import java.util.Scanner;

class Faktorial14 {

    public int faktorialBF(int n) {

        int hasil = 1;

        for (int i = 1; i <= n; i++) {

            hasil *= i;

        }

        return hasil;

    }

    public int faktorialDC(int n) {

        if (n == 1 || n == 0) {

            return 1;

        }

        return n * faktorialDC(n - 1);

    }

}

public class MainFaktorial {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Masukkan angka: ");

        int n = sc.nextInt();

        Faktorial14 f = new Faktorial14();

        System.out.println("Faktorial BF: " + f.faktorialBF(n));

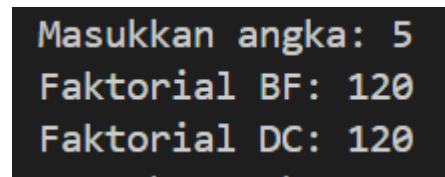
        System.out.println("Faktorial DC: " + f.faktorialDC(n));

        sc.close();

    }

}
```

5.2.1 Verifikasi Hasil Percobaan



```
Masukkan angka: 5
Faktorial BF: 120
Faktorial DC: 120
```

5.2.2 Pertanyaan

1. Perbedaan If dan else

- Bagian if digunakan untuk menghentikan rekursi saat mencapai angka 0 atau 1, karena faktorial dari kedua angka itu adalah 1.
- else digunakan untuk melanjutkan perhitungan dengan cara memecah masalah menjadi lebih kecil, yaitu mengalikan angka sekarang dengan faktorial dari angka sebelumnya (n-1).

2. Bisa, dengan menggunakan while maupun do while

```
public int faktorialBF(int n) {
    int hasil = 1;
    int i = 1;
    while (i <= n) {
        hasil *= i;
        i++;
    }
    return hasil;
}
```

3. Perbedaan fakto *= i; dan int fakto = n * faktorialDC(n-1);

- fakto *= i; dipakai dalam cara Brute Force, di mana faktorial dihitung dengan mengalikan angka satu per satu dalam loop.
- int fakto = n * faktorialDC(n-1); dipakai dalam Divide and Conquer, yang memecah masalah dengan memanggil dirinya sendiri secara rekursif sampai mencapai angka terkecil (base case).

4. Kesimpulan cara kerja method faktorialBF() dan faktorialDC()

- faktorialBF() Menghitung dengan perulangan, lebih cepat karena langsung dihitung tanpa memanggil fungsi lain berulang kali.
- faktorialDC() Menggunakan rekursi, jadi lebih mudah dibaca dan lebih elegan, tapi butuh lebih banyak memori karena harus menyimpan banyak pemanggilan fungsi dalam antrian.

5.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan DivideandConquer

```
import java.util.Scanner;

class Pangkat14 {
    int nilai, pangkat;

    public Pangkat14(int nilai, int pangkat) {
        this.nilai = nilai;
        this.pangkat = pangkat;
    }

    int pangkatBF (int a, int nilai) {
        int hasil = 1;
        for (int i = 0; i < nilai; i++) {
            hasil *= a;
        }
        return hasil;
    }

    int pangkatDC (int a, int nilai) {
        if (nilai == 1) {
            return a;
        } else {
            if (nilai % 2 == 1) {
                return a * pangkatDC(a, nilai/2) * pangkatDC(a, nilai/2);
            } else {
                return pangkatDC(a, nilai/2) * pangkatDC(a, nilai/2);
            }
        }
    }
}

public class MainPangkat {
    public static void main(String[] args) {
        Scanner input = new Scanner (System.in);
```

```

        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();

        Pangkat14[] png = new Pangkat14[elemen];
        for(int i=0; i<elemen; i++) {
            System.out.print("Masukkan nilai yang dipangkatkan ke-" + (i+1) + ": ");
            int nilai = input.nextInt();
            System.out.print("Masukkan nilai pangkat ke-" + (i+1) + ": ");
            int pangkat = input.nextInt();
            png[i] = new Pangkat14(nilai, pangkat);
        }

        System.out.println("HASIL PANGKAT BRUTFORCE:");
        for (Pangkat14 p : png) {
            System.out.println(p.nilai + " pangkat " + p.pangkat + " adalah: " +
                p.pangkatBF(p.nilai, p.pangkat));
        }

        System.out.println("HASIL PANGKAT DIVIDE AND CONQUER:");
        for (Pangkat14 p : png) {
            System.out.println(p.nilai + " pangkat " + p.pangkat + " adalah: " +
                p.pangkatDC(p.nilai, p.pangkat));
        }
    }
}

```

5.3.1 Verifikasi hasil percobaan

```

Masukkan jumlah elemen: 3
Masukkan nilai yang dipangkatkan ke-1: 2
Masukkan nilai pangkat ke-1: 3
Masukkan nilai yang dipangkatkan ke-2: 4
Masukkan nilai pangkat ke-2: 5
Masukkan nilai yang dipangkatkan ke-3: 6
Masukkan nilai pangkat ke-3: 7
HASIL PANGKAT BRUTFORCE:
2 pangkat 3 adalah: 8
4 pangkat 5 adalah: 1024
6 pangkat 7 adalah: 279936
HASIL PANGKAT DIVIDE AND CONQUER:
2 pangkat 3 adalah: 8
4 pangkat 5 adalah: 1024
6 pangkat 7 adalah: 279936

```

5.3.2 Pertanyaan

1. perbedaan method pangkatBF() dan pangkatDC()
 - pangkatBF() Menghitung pangkat dengan cara mengalikan angka berulang kali di dalam loop. Sederhana, tapi kalau angkanya besar, bisa butuh banyak waktu.
 - pangkatDC() Menggunakan rekursi untuk membagi masalah jadi lebih kecil. Kalau pangkatnya genap, cukup kalikan hasil rekursinya sendiri (hemat perkalian). Kalau pangkatnya ganjil, perlu dikali satu angka lagi.
- 2.

```
if (nilai == 1) {  
    return a;  
} else {  
    if(nilai%2==1){  
        return a * pangkatDC(a, nilai/2) * pangkatDC(a, nilai/2);  
    } else {  
        return pangkatDC(a, nilai/2) * pangkatDC(a, nilai/2);  
    }  
}
```

3.

```
int pangkatBF() {  
    int hasil = 1;  
    for (int i = 0; i < this.pangkat; i++) {  
        hasil *= this.nilai;  
    }  
    return hasil;  
}
```

Bisa saja tanpa menggunakan parameter Namun, jika ingin lebih fleksibel (misalnya, pangkat angka yang berbeda tanpa membuat objek baru), parameter tetap bisa dipertahankan.

4. Kesimpulan cara kerja pangkatBF() dan pangkatDC()
 - pangkatBF() Menggunakan loop untuk melakukan perkalian berulang kali. Cara ini mudah dimengerti, tetapi kurang efisien untuk pangkat yang besar.
 - pangkatDC() Menggunakan rekursi dan membagi masalah menjadi lebih kecil, sehingga jumlah perkalian berkurang drastis, membuatnya lebih cepat dibanding pangkatBF().

5.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

```
import java.util.Scanner;

class sum14 {
    double keuntungan[];

    sum14(int el) {
        keuntungan = new double[el];
    }

    double totalBF() {
        double total = 0;
        for (int i = 0; i < keuntungan.length; i++) {
            total += keuntungan[i];
        }
        return total;
    }

    double totalDC(double arr[], int l, int r) {
        if (l == r) {
            return arr[l];
        }

        int mid = (l + r) / 2;
        double lsum = totalDC(arr, l, mid);
        double rsum = totalDC(arr, mid + 1, r);
        return lsum + rsum;
    }
}

public class mainsum14 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();
        sum14 sm = new sum14(elemen);
        for (int i = 0; i < elemen; i++) {
            System.out.print("Masukkan keuntungan ke-" + (i + 1) + ": ");
            sm.keuntungan[i] = input.nextDouble();
        }
    }
}
```

```

    }

    System.out.println("Total Keuntungan menggunakan BruteForce: " +
sm.totalBF());

    System.out.println("Total Keuntungan menggunakan Divide and Conquer: " +
sm.totalDC(sm.keuntungan, 0, elemen - 1));

    input.close();

    }
}

```

5.4.1 Verifikasi Hasil Percobaan

```

Masukkan jumlah elemen: 5
Masukkan keuntungan ke-1: 10
Masukkan keuntungan ke-2: 20
Masukkan keuntungan ke-3: 30
Masukkan keuntungan ke-4: 40
Masukkan keuntungan ke-5: 50
Total Keuntungan menggunakan BruteForce: 150.0
Total Keuntungan menggunakan Divide and Conquer: 150.0

```

5.4.2 pertanyaan

1. mengapa perlu ada variable mid pada method TotalDC() karena
 - mid digunakan untuk membagi array menjadi dua bagian.
 - Karena algoritma Divide and Conquer bekerja dengan cara membagi masalah besar menjadi lebih kecil, kita perlu tahu titik tengah agar bisa memisahkan bagian kiri dan kanan.
2. lsum menghitung jumlah dari bagian kiri array, sedangkan rsum menghitung jumlah dari bagian kanan array.
3. Karena jika tanpa menjumlahkan lsum dan rsum, kita hanya mendapatkan sebagian hasil, bukan total seluruh array.
4. Base case terjadi ketika hanya ada satu elemen dalam array, yaitu saat left == right.
5. Kesimpulan cara kerja totalDC()
 - **Divide:** Array dibagi menjadi dua bagian sampai hanya tersisa satu elemen per bagian.
 - **Conquer:** Nilai setiap bagian dijumlahkan secara rekursif.
 - **Combine:** Hasil dari dua bagian digabungkan untuk mendapatkan total keseluruhan.

4.5 Latihan Praktikum

```
class Nilai14 {

    String nama;

    String nim;

    int tahunMasuk;

    int nilaiUTS;

    int nilaiUAS;

    public Nilai14(String nama, String nim, int tahunMasuk, int nilaiUTS, int
nilaiUAS) {

        this.nama = nama;

        this.nim = nim;

        this.tahunMasuk = tahunMasuk;

        this.nilaiUTS = nilaiUTS;

        this.nilaiUAS = nilaiUAS;

    }

}

public class MainNilai14 {

    public static void main(String[] args) {

        Nilai14[] mahasiswa = {

            new Nilai14("Ahmad", "220101001", 2022, 78, 82),

            new Nilai14("Budi", "220101002", 2022, 85, 88),

            new Nilai14("Cindy", "220101003", 2021, 90, 87),

            new Nilai14("Dian", "220101004", 2021, 76, 79),

            new Nilai14("Eko", "220101005", 2023, 92, 95),

            new Nilai14("Fajar", "220101006", 2020, 88, 85),

            new Nilai14("Gina", "220101007", 2023, 80, 83),

            new Nilai14("Hadi", "220101008", 2020, 82, 84)

        };

        int nilaiUTSTertinggi = findMaxUTS(mahasiswa, 0, mahasiswa.length - 1);

        int nilaiUTSTerendah = findMinUTS(mahasiswa, 0, mahasiswa.length - 1);

        double rataRataUAS = calculateAverageUAS(mahasiswa);
```

```

        System.out.println("Nilai UTS tertinggi: " + nilaiUTSTertinggi);
        System.out.println("Nilai UTS terendah: " + nilaiUTSTerendah);
        System.out.println("Rata-rata nilai UAS: " + rataRataUAS);
    }

    public static int findMaxUTS(Nilai14[] arr, int low, int high) {
        if (low == high) {
            return arr[low].nilaiUTS;
        }

        int mid = (low + high) / 2;
        int leftMax = findMaxUTS(arr, low, mid);
        int rightMax = findMaxUTS(arr, mid + 1, high);

        return Math.max(leftMax, rightMax);
    }

    public static int findMinUTS(Nilai14[] arr, int low, int high) {
        if (low == high) {
            return arr[low].nilaiUTS;
        }

        int mid = (low + high) / 2;
        int leftMin = findMinUTS(arr, low, mid);
        int rightMin = findMinUTS(arr, mid + 1, high);

        return Math.min(leftMin, rightMin);
    }

    public static double calculateAverageUAS(Nilai14[] arr) {
        int total = 0;
        for (Nilai14 nilai : arr) {
            total += nilai.nilaiUAS;
        }
        return (double) total / arr.length;
    }
}

```

4.5.1 Verifikasi Hasil Praktikum

```
Nilai UTS tertinggi: 92  
Nilai UTS terendah: 76  
Rata-rata nilai UAS: 85.375
```

4.5.2 Link GitHub

<https://github.com/ianmen10/semester2.git>

